



AGILE SOFTWARE DEVELOPMENT

Lee Rodger - 1809707

Abstract

This document outline principles of agile software development methodologies and compares them with the traits associated with traditional software development. The techniques used within the methodologies will be discussed and analysed, highlighting the processes which are carried out.

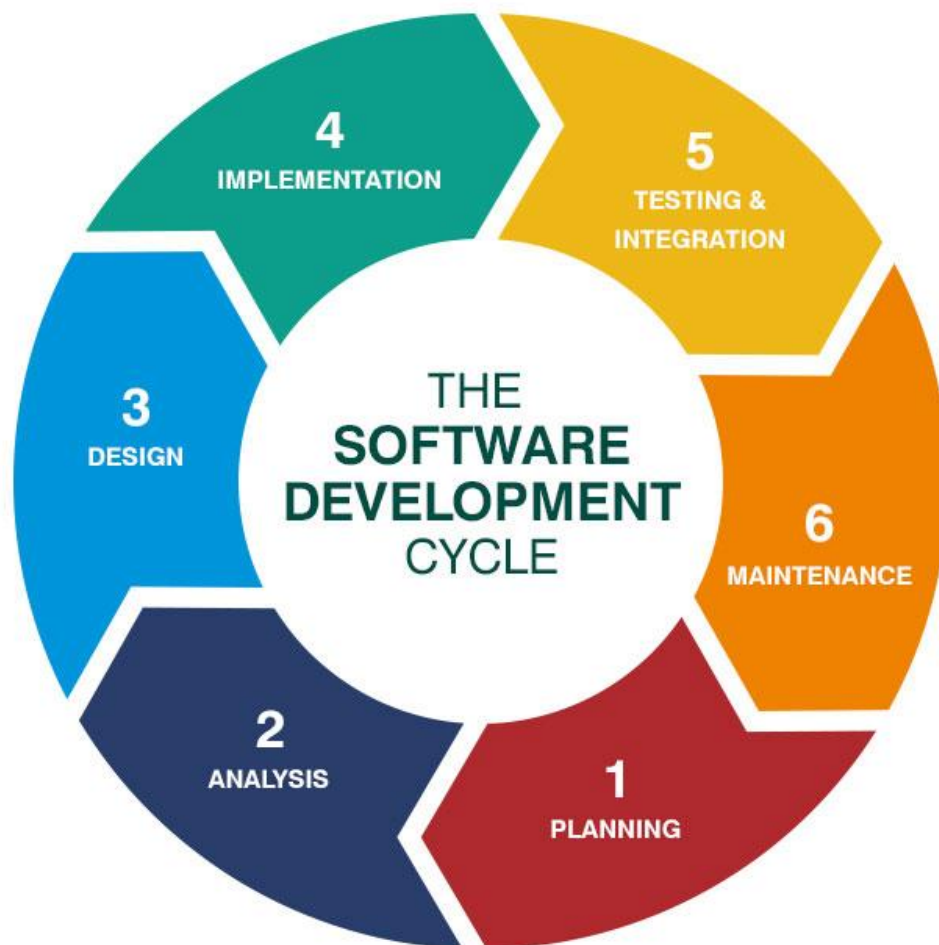
Contents

1 – Introduction	2
2 – The Agile Manifesto Principles.....	3
3 – Traditional Software Development Versus Agile	3
4 – Scrum	4
5 – Version Control and GIT	5
6 – How agile is used in the DVLA.....	6
7 – Summary and Conclusion.....	7
References	8

1 – Introduction

Agile software development methods have revolutionised the way in which software development industries work. It allows for an iterative development process and rapid prototyping [1] where multiple teams can work together to quickly create a minimum viable product whereby the customer can monitor its development and be in regular communication with the team to make sure they get what they want by the end.

Most developers will follow a version of the software development life cycle since it defines the tasks which must be performed at each stage. It is a method of quality control and encourage the team to all be aware of the current stage of development. Below is an example of the lifecycle [Figure 1.1].



[Figure 1.1] – The software development lifecycle, taken from Tricia Hussung [9].

These stages occur consistently throughout software development, and the cycle gives an indication to how agile methodologies work within an organisation.

2 – The Agile Manifesto Principles

The agile manifesto contains all relevant information pertinent to the development approach. Importantly, it outlines the 12 principles used which outline the goals agile as a whole [2]. There are a number of the principles which outline some of the most important features of agile software development.

Firstly, the highest priority of agile development is to satisfy the customer by delivering early and continuous software iterations. This is especially important for the developers and the customer. The team is in regular communication with the customer allowing them to achieve the exact outcome desired in less time with fewer chances of mistakes. This also saves the customer money as they are able to pay for further development until they are happy with the product.

Agile development means that developers can be very open to a change in requirements, no matter how far into development they are. Since communication with the customer is regular, they are able to communicate changes and implement them quickly. This removes the risk of mistakes further down the line and the customer will remain satisfied.

The process of the agile methodology encourages teams to deliver working software as quickly as possible. The teams will initially work towards creating a minimum viable product. Once this is achieved, they can show the customer this prototype which will be operational for all of the necessary requirements laid out by the customer. From here, the team can continue to develop and implement more features, increasing the capability of already working software.

3 – Traditional Software Development Versus Agile

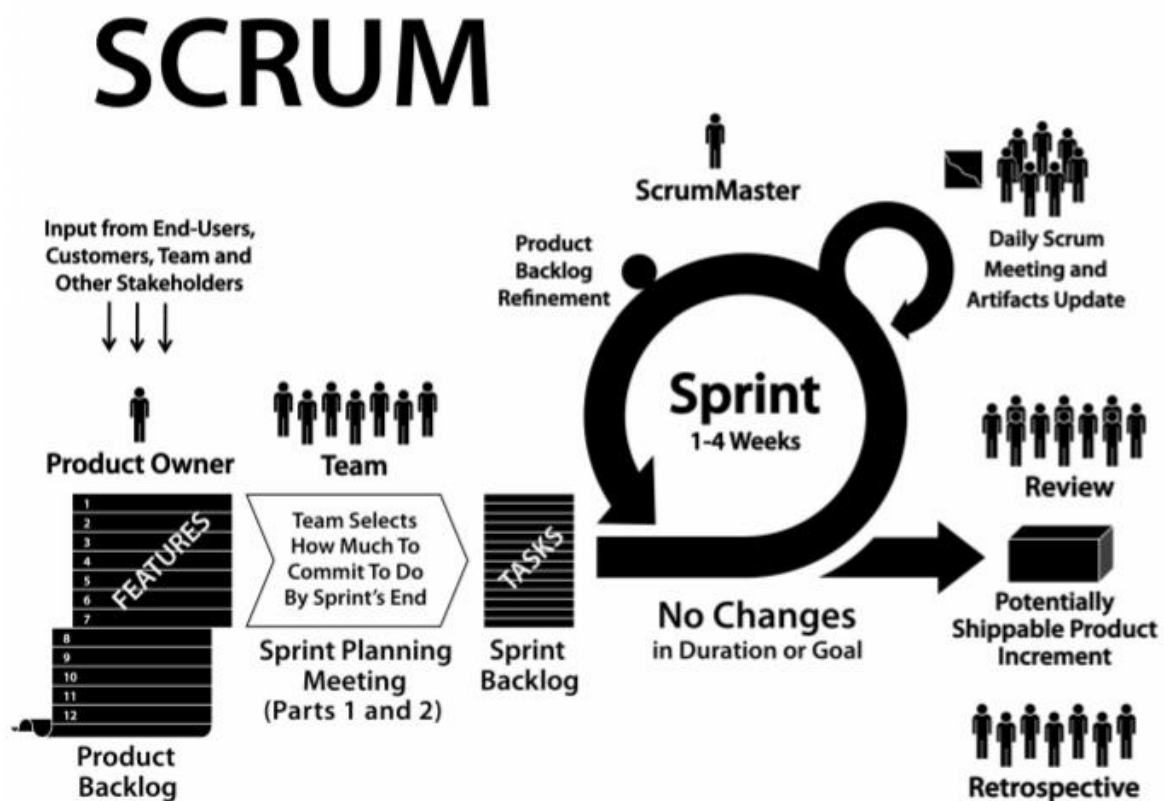
Traditional methodologies for software development are typically very structured and do not adapt well to change or any issues which may arise. In a traditional methodology such as waterfall, there are a number of large stages which are always carried out the same way. Typically, it begins with an extensive phase of planning, whereby the final goal/end-product has been considered, designed and heavily documented [3]. Only once the stakeholders have reviewed the plans does the team begin to work on project, following the design exactly.

This however can create a number of issues further down the development line. The product owner will usually never see the product until it has reached the end of development. This means that the product owner could turn around and say that the final outcome is not what they wanted. The cost of redesigning and changing a product at this stage would be considerably large and very time consuming as the documentation would need to be altered and redesigned to account for the new changes. This approach also cannot account for mistakes and failures occurring along the way. If a problem arises in the design, it is also a very costly and time consuming process whereby the design needs to be re-altered and approved by the product owner. Traditional methodologies do not adapt well to change and therefore do not always fit well into an industry whereby continuous iterations and development is suited. This is where agile becomes a necessity within most development teams.

The agile development process focuses around delivering a working product quickly and iteratively to the product owner. Whilst there are a number of different methodologies to choose from, they all follow the same basic principles and can be incorporated together to suit the development team in question. Agile methodologies can also help the product owner to feel more confident as they only need to pay for the amount of work they want to and get to see a product very quickly which will meet the minimum specification.

4 – Scrum

A common agile methodology is scrum. It is an iterative and incremental framework which focuses on delivering work in 'sprints' [3]. A sprint is a period of work whereby the development team will work on sections of the product, and is led by the 'scrum master'. This usually last for around 2 weeks, but can be altered to suit any team. All of the requirements are written in the form of 'user stories'. These are structured in the form of "As a (User/Customer/Administrator etc.), I want to (Something that can be done with the product) so that (Reason for doing that action)". These are put into the 'project backlog' which contains all of the user storied that have been created for the project. From this backlog, stories are picked to go into a sprint which is what the development team will work on for that cycle. The ultimate goal is to complete all of the stories in that sprint by the end. At the end of each sprint, there will be a review, often involving the product owner so that they can see the progress, allowing for changes to be made if necessary. Once the review is complete, the team conduct a retrospective, analysing what went well, what went wrong and what they could do to improve next time. After this, the cycle begins again with the planning for the next sprint. This diagram [Figure 4.1] demonstrates the scrum lifecycle clearly.



[Figure 4.1] – Diagram of the scrum lifecycle, taken from "The Scrum Papers [3]".

Scrum in general is very beneficial for developers because it allows for quick development that is very adaptable to a change in requirements [4]. This means that issues can be rectified very quickly and with minimal impact on the time frame for the project.

Within scrum, each user story is given a weighting of complexity using the Fibonacci scale. Each member of the development squad will vote at the same time on their estimated level of complexity for each story during the sprint planning stage. The member with the highest vote and the member

with the lowest vote will then discuss their reasons for the score they have given. The squad continue to re-vote until they can establish a score which they all agree upon and apply that to the story. This score gives each member an idea of how complex a certain story might be to achieve. This does not necessarily reflect the time it will take to complete that story, but is an indicator of the level of knowledge or research required to successfully implement it.

Each day of the sprint, the squad conduct a 'stand-up' whereby each member stands up and in turn discuss their progress on the current sprint. Each member states what they did yesterday, what they plan to do that day, and if there are any blockers which may cause issues for them to make progress. This gives the whole squad an idea of where everyone is at and what issues need to be resolved. The reason the team stands up is to encourage people to not go into great detail when discussing their part. Sitting down discussing with a team usually leads to long conversations rather than work being carried out. Standing up means that the team will conduct the meeting in as little time as possible, with only the relevant information being conveyed.

5 – Version Control and GIT

Software development teams often rely on some sort of version control software to keep track of their work and be able to easily work collaboratively on the same project at the same time. Most organisations will use software such as GIT in order to achieve this goal.

GIT is a version-control software which works with both remote and local repositories [5]. The whole project will be contained on a remote repository which all members of the team have access to. This in theory allows teams to work from anywhere in the world if necessary on the same project.

GIT allows a team member to pull down the latest version of the project and create a 'branch', allowing them to develop a new feature without affecting all of the current code in the 'master' branch, containing all of the up to date code. This branch is also accessible by all other team members once it is pushed back up to the remote repository. Once it is pushed, the developer creates a 'pull request' whereby two or more members of the squad will review their work and can approve it if it is ready to be merged with the master branch, updating the code for everyone else the next time they make a pull. Usually it is the scrum master who makes the merge with the project.

The use of version control also allows the team to 'roll back', allowing them to return to a previous version of the project if someone goes wrong. This means that something which is implemented that causes problems can be easily removed and adjusted before being put back into the main project.

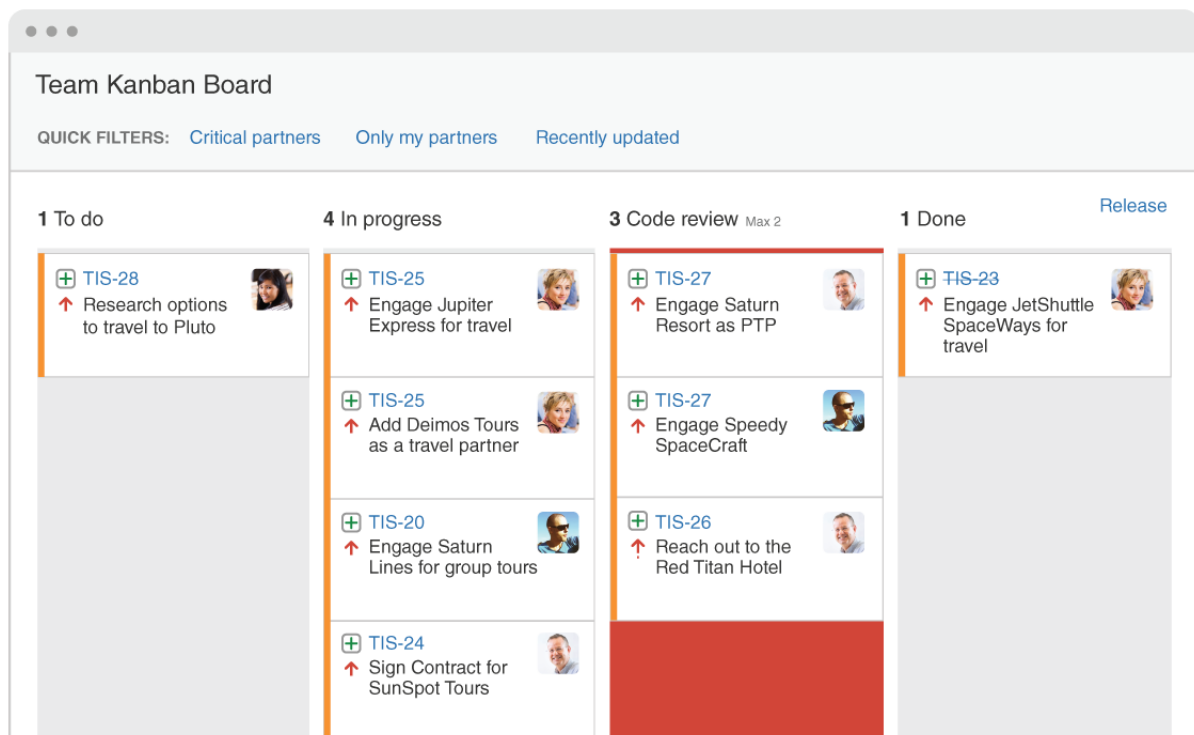
This aids development as all work is kept safe and all members can work on their stories from any location and commit then push their changes back up to the remote repository. This is ideal for agile development squads because it means that teams in different locations can effectively work together and keep the project constantly up-to-date and ready for new features.

6 – How agile is used in the DVLA

The DVLA work entirely using agile development methodologies. While the primary focus is on the scrum methodology, they also use a combination of Kanban and Lean too which works well for the development squads.

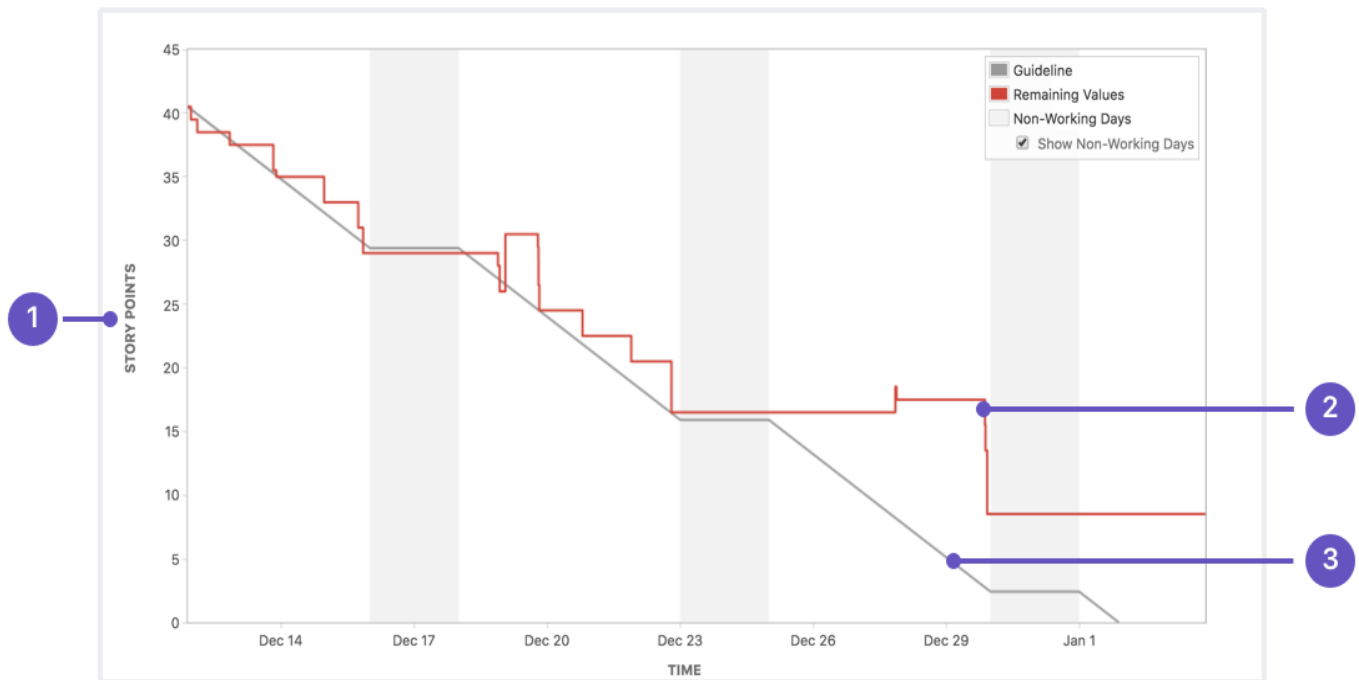
Kanban is an agile methodology which requires “real-time communication and capacity and full transparency of work” [6]. It dates back as far as the late 1940s and was developed as a means of maximum production potential within Toyota. The focus is on finding the optimal working capacity. This is a huge focus within the DVLA.

Primarily they use a feature of Kanban called the ‘Kanban board’ [7]. This feature allows the squad to have a visual layout of where stories are along the development route. Stories in the current sprint are placed on a board into columns representing their current state. The typical states are ‘To do’, ‘In Progress’, ‘Ready for Review’, ‘Ready for Testing’ and ‘Done’. Stories move along the board when they are in the next stage of development. It is usually the scrum master who updates this board during the sprint stand-up. An example of a Kanban board can be seen below [Figure 6.1].



[Figure 6.1] – Example of a Kanban board taken from Atlassian Agile Coach [6].

The boards' stages can vary making them ideal for any squad who needs to make adjustments to fit better to their needs. This is also combined with what's known as a 'burn-down chart'. This displays a timeline of the current sprint with a line representing the number of stories left to complete. In an ideal world, this would reach 0 by the end of the sprint. However, teams will usually overestimate how much work they can take on so that they do not complete work early and have little to do before the next sprint.



[Figure 6.2] – Example of a Burndown chart taken from Atlassian Agile Coach [8].

The above chart shows a grey line representing the ultimate goal and stages which the squad should be at by certain dates. The red line represents the current progress made. The line moves down when a story is completed but can also move up if another story is added to the backlog. This often happens when it is realised that another feature must also be implemented at this time, usually linking to an already existing story.

7 – Summary and Conclusion

The software development industry in general is steering towards the agile development methodologies. Most industries are now creating software which needs to be updated and managed throughout its use which means that having an iterative development process which continually updates a product fits in well.

Some businesses will continue to use traditional methodologies as this can fit better for certain products. For example, a company developing software components for vehicles would benefit better from rigorous planning using the waterfall method as they are setting out to achieve a fixed goal with every detail planned out prior. There will be no need to continue developing this product after its completion. However, there is still increasing demand for agile development for most organisations who want to instill faith in the product owners and develop working software in a timely fashion with visible and usable results becoming available early in the development process.

References

- [1] P. K. B. Guarav Kumar, “Impact of Agile Methodology on Software Development Process,” *International Journal of Computer Technology and Electronics Engineering (IJCTEE)*, 2012.
- [2] Agile Alliance, “12 Principles Behind the Agile Manifesto,” Agile Alliance, February 2001. [Online]. Available: <https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/>. [Accessed 11 January 2019].
- [3] J. Sutherland, *The Scrum Papers*, Cambridge, 2012.
- [4] J. a. S. K. Sutherland, *The Scrum Guide*, 2017.
- [5] git, “git - About,” 2019. [Online]. Available: <https://git-scm.com/about>. [Accessed 11 January 2019].
- [6] Atlassian Agile Coach, “Kanban,” Atlassian, 2019. [Online]. Available: <https://www.atlassian.com/agile/kanban>. [Accessed 11 January 2019].
- [7] Atlassian, “WIp-Limits,” 2019. [Online]. Available: <https://www.atlassian.com/agile/kanban/wip-limits>. [Accessed 11 January 2019].
- [8] Atlassian Agile Coach, “Burndown Charts,” Atlassian, 2019. [Online]. Available: <https://www.atlassian.com/agile/tutorials/burndown-charts>. [Accessed 11 January 2019].
- [9] T. Hussung, “What Is the Software Development Life Cycle?,” Husson University Online, 10 March 2016. [Online]. Available: <https://online.husson.edu/software-development-cycle/>. [Accessed 11 January 2019].